Datorsystemteknik

för D2, ICT2, E3 och Mek3

Nicholas Wickström

Högskolan i Halmstad

Sverige

Outline

- Föregående fl.
- Datastrukturer i kärnan
- Funktioner i kärnan
- Extra funktioner

Typer

```
typedef int          exception;
typedef int          bool;
typedef unsigned int uint;

#define FAIL 0
#define SUCCESS 1
#define OK 1
#define DEADLINE_REACHED 0
#define NOT_EMPTY 0
```

Task control block, TCB

```
/* Task Control Block, TCB */
typedef struct _TCB
{
  void (*PC)(); /* Program counter */
  uint *SP; /* Stack pointer */
  uint Context[CONTEXT_SIZE]; /* TCB context */
  uint StackSeg[STACK_SIZE]; /* TCB stack */
  uint DeadLine; /* The deadline of the TCB */
} TCB;
```

Message Objects

```
/* Message objects */
typedef struct msgobj {
   void *pData;
   exception Status; /*{OK,FAIL,DEADLINE_REACHED}*/
   struct l_obj *pBlock; /* NULL for non-block msg */
   struct msgobj *pPrevious;
   struct msgobj *pNext;
} msg;
```

Mailboxes

```
/* Mailbox structure */
typedef struct _mailbox {
   msg *pHead;
   msg *pTail;
   int nDataSize; /* Size of data */
   int nMaxMessages; /* Max number of list objects */
   int nMessages; /* Number of senders/receivers */
   int nBlockedMsg; /* Number of blocked senders/receive
} mailbox;
```

Generic List Objects

```
/* Generic list item */
typedef struct l_obj {
    TCB         *pTask; /* Encapsulated TCB */
    uint        nTCnt; /* Waiting time in Timerlist */
    msg         *pMessage; /* Address of current message
    struct l_obj  *pPrevious;
    struct l_obj  *pNext;
} listobj;
```

Generic list

```
/* Generic list */
typedef struct _list {
    listobj *pHead;
    listobj *pTail;
} list;
```

SaveContext

```
extern void SaveContext( void );
/* Stores registers in TCB (pointed to by Running) */
```

1) Blockera timer interrupt
2) Fixa stacken (pga funktionsanropet)
3) Lagra PC, SP, register i TCB
4) Återställ timer interrupt

LoadContext

```
extern void LoadContext( void );
/* Load registers from TCB (pointed to by Running) */
```

1) Blockera timer interrupt
2) Sätt PC till TCBs PC, SP till TCBs SP
3) Återställ register från TCBs Context
4) Återställ timer interrupt

timer0_isr

Denna funktionen anropas från avbrottsvektorn. Anropen sker varje "tick".

1) Spara context (i Running)
2) Anropar C-funktionen TimerInt()
3) Återställ context (från Running)

kernel_hwdep.c

Dessa funktionerna är hårdvaruberoende.

```
uint set_isr( uint isr_level )  /* set isr on/off, retur
void timer0_start( void ) /* Setup of timer */
void EnableTimerInterrupt( void )
/* Setup of timer int */
isr_off() /* Disable ints */
isr_on() /* Enable ints */
```